Package: naturecounts (via r-universe)

August 31, 2024

Type Package

Title Access and download data on plant and animal populations from NatureCounts

Version 0.4.1

Description Access and download data on plant and animal populations from various databases through NatureCounts, a service managed by Bird Studies Canada.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.1.2)

Imports askpass (>= 1.1), DBI (>= 1.0.0), dbplyr (>= 1.4.4), dplyr (>= 1.1.0), ggplot2 (>= 3.4.0), httr (>= 1.4.0), jsonlite (>= 1.6), lubridate (>= 1.7.4), magrittr, memoise (>= 2.0.1), purrr (>= 0.3.2), rlang (>= 0.1.2), RSQLite (>= 2.1.1), stringi (>= 1.2.4), stringr (>= 1.3.1), tibblify (>= 0.3.0), tidyr (>= 0.8.2), tidyselect (>= 1.0.0), units (>= 0.8.1), wk (>= 0.7.1)

Suggests ggspatial, here, knitr, mapview, rnaturalearth, roxygen2, rmarkdown, sf (>= 1.0-9), spelling, testthat, vdiffr

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/BirdsCanada/naturecounts, https://naturecounts.ca, https://birdscanada.github.io/naturecounts/

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://steffilazerte.r-universe.dev

RemoteUrl https://github.com/BirdStudiesCanada/naturecounts

RemoteRef HEAD

RemoteSha 6d368c47524ea7236a3ff523ab799be582873727

1

Contents

---

naturecounts-package   *Access and download data on plant and animal populations from Na-*
tureCounts

Description

naturecounts is an R package for accessing and downloading data on plant and animal populations from various databases through NatureCounts, a service managed by Bird Studies Canada.

Details

See the vignettes (`vignettes(package = "naturecounts")`, or at https://birdscanada.github.io/naturecounts for getting started!

Author(s)

Maintainer: Denis Lepage `<dlepage@birdscanada.org>`

Authors:

• Steffi LaZerte `<steffi@steffi.ca>`

See Also

Useful links:

- <https://github.com/BirdsCanada/naturecounts>
- <https://naturecounts.ca>
- <https://birdscanada.github.io/naturecounts/>

| bcch | *Example black-capped chickadee data* |

Description

An example of black-capped chickadee data downloaded from NatureCounts

Usage

```
bcch
```

Format

A data frame with 160 rows and 57 variables:

| cosewic_plot | *Plot COSEWIC IAO and EOO* |

Description

Creates a plot of COSEWIC ranges for illustration and checking.

Usage

```
cosewic_plot(
  ranges,
  points = NULL,
  grid = NULL,
  map = NULL,
  species = "species_id",
  title = ""
)
```

Arguments

ranges	List. Output of `cosewic_ranges()` with `spatial = TRUE`.
points	Data frame. Optional naturecounts data used to compute ranges. Raw data points will be added to the plot if provided.
grid	sf data frame. Optional grid over which to summarize IAO values (useful for species with many points over a broad distribution).
map	sf data frame. Optional base map over which to plot the values.
species	Character. Name of the column containing species identification.
title	Character. Optional title to add to the map. Can be a named by species vector to supply different titles for different species.

Value

ggplot2 map

Examples

```
r <- cosewic_ranges(bcch)
cosewic_plot(r)
cosewic_plot(r, points = bcch)
cosewic_plot(r, grid = grid_canada(50), map = map_canada(),
             title = "Black-capped chickadees")

m <- rbind(bcch, hofi)
r <- cosewic_ranges(m)
cosewic_plot(r)
cosewic_plot(r, points = m)
p <- cosewic_plot(r, grid = grid_canada(50), map = map_canada(),
                  title = c("14280" = "Black-capped chickadees",
                            "20350" = "House Finches"))
p[[1]]
p[[2]]
```

---

cosewic_ranges   *Calculate COSEWIC IAO and EOO*

Description

The COSEWIC Index of Area of Occupancy (IAO; also called Area of Occupancy, AOO by the IUCN) and Extent of Occurrence (EOO; IUCN as well) are metrics used to support status assessments for potentially endangered species.

Usage

```
cosewic_ranges(
  df_db,
  record = "record_id",
  coord_lon = "longitude",
  coord_lat = "latitude",
  species = "species_id",
  iao_grid_size_km = 2,
  eoo_p = 0.95,
  filter_unique = FALSE,
  spatial = TRUE
)
```

Arguments

`df_db`	Either data frame or a connection to database with `naturecounts` table.
`record`	Character. Name of the column containing record identification.
`coord_lon`	Character. Name of the column containing longitude.
`coord_lat`	Character. Name of the column containing latitude.
`species`	Character. Name of the column containing species identification.
`iao_grid_size_km`	Numeric. Size of grid (km) to use when calculating IAO. Default is COSEWIC requirement (2). Use caution if changing.
`eoo_p`	Numeric. The percentile to calculate the convex hull over. Defaults to 0.95 for a 95% convex hull to ensure outlier points do not artificially inflate the EOO. Note that for a final COSEWIC report, this may not be appropriate. Set to 1 to include all points.
`filter_unique`	Logical. Whether to filter observations to unique locations. Use this only if there are too many data points to work with. This changes the nature of what an observation is, and also may bias EOO calculations if using less than 100% of points (see `eoo_p`).
`spatial`	Logical. Whether to return sf spatial objects showing calculations. If `FALSE` (the default) returns a data frame with the IAO and EOO values. If `TRUE` returns a list of objects with both the values and the spatial grid/polygons.

Details

Note that the while the IUCN calls this metric AOO, in COSEWIC, AOO is actually a different measure, the *biological* area of occupancy. See the Distribution section in 'Instructions for preparing COSEWIC status reports' for more details.

By default the EOO is calculated only using the inner 95% of points (based on distance to the centroid). This is to ensure that a first-pass of the EOO does not reject a species from consideration if there are any outlier observations. However, for a final COSEWIC assessment report, it is likely better to carefully explore the data to ensure there are no outliers and then use the full data set (i.e. set eoo_p = 1).

The IAO is calculated by first assessing large grids (10x large than the specified size). Only then are smaller grids created within large grid cells containing observations. This speeds up the process by avoiding the creation of grids in areas where there are no observations. This means that the plots and spatial objects may not have grids over large areas lacking observations. See examples.

Details on how IAO and EOO are calculated and used

- COSEWIC - Guidelines for use of the Index of Area of Occupancy in COSEWIC Assessments
- COSEWIC - Instructions for preparing COSEWIC status reports
- COSEWIC - Table 2 COSEWIC quantitative criteria and guidelines for the status assessment of Wildlife Species

Value

Summarized data frame (ranges) or list containing `ranges`, a summarized data frame, and `spatial`, a list of two spatial data frames.

`ranges` contains columns

- `n_records_total` - Total number of records used to create ranges
- `min_record` - Minimum number of records within IAO cells
- `max_record` - Maximum number of records within IAO cells
- `median_record` - Median number of records within IAO cells
- `grid_size_km` - IAO cell size (area is this squared)
- `n_occupied` - Number of IAO cells with at least one record
- `iao` - IAO value (`grid_size_km^2 * n_occupied`)
- `eoo_pXX` - EOO area calculated with a convex hull at percentile eoo_p (e.g., 95%)

`spatial` contains spatial data frames

- `iao_sf` - Polygons of the IAO grids with the `n_records` per cell
- `eoo_sf` - Polygon of the Convex Hull at percentile eoo_p

Examples

```
# Using the included, test data on black-capped chickadees

bcch # look at the data

r <- cosewic_ranges(bcch)
r <- cosewic_ranges(bcch, spatial = FALSE)

# Calculate for multiple species
mult <- rbind(bcch, hofi)
r <- cosewic_ranges(mult)
r <- cosewic_ranges(mult, spatial = FALSE)
```

---

format_dates *Add date and day-of-year field/columns to data*

Description

Creates and adds columns `date` and `doy` (day-of-year) to the data source (either data frame or database table `naturecounts`).

Usage

```
format_dates(df_db, overwrite = FALSE)
```

Arguments

df_db	Either data frame or a connection to database with `naturecounts` table. Must have fields/columns of `survey_year`, `survey_month`, `survey_day`
overwrite	Logical. Overwrite existing columns `date` and/or `doy`?

Value

If `df_db` was a data frame, return a data frame with new columns `date` and `doy`. Otherwise return database connection.

Examples

```
bcch_with_dates <- format_dates(bcch)
```

---

format_zero_fill *Zero-fill data*

Description

Zero-fill the species presence data by adding zero observation counts (absences) data to an existing naturecounts dataset.

Usage

```
format_zero_fill(
  df_db,
  by = "SamplingEventIdentifier",
  species = "all",
  fill = "ObservationCount",
  extra_species = NULL,
  extra_event = NULL,
  warn = TRUE,
  verbose = TRUE
)
```

Arguments

`df_db`	Either data frame or a connection to database with `naturecounts` table (a data frame is returned).
`by`	Character vector. By default, "SamplingEventIdentifier" or a vector of specific column names to fill by (see details)
`species`	Character vector. Either "all", for species in the data, or a vector of species ID codes to fill in.
`fill`	Character. The column name to fill in. Defaults to "ObservationCount".
`extra_species`	Character vector. Extra columns/fields uniquely associated with `species_id` to keep in the data (all columns not in by, `species`, `fill`, `extra_species`, or `extra_event` will be omitted from the result).
`extra_event`	Character vector. Extra columns/fields uniquely associated with the Sampling Event (the field defined by by) to keep in the data (all columns not in by, `species`, `fill`, `extra_species`, or `extra_event`) will be omitted from the result).
`warn`	Logical. If TRUE, stop zero-filling if >100 species and >1000 unique sampling events. If FALSE, ignore and proceed.
`verbose`	Logical. Show messages?

Details

by refers to the combination of columns which are used to detect missing values. By default `SamplingEventIdentifier` is used. Otherwise users can specify their own combination of columns.

If `species` is supplied, all records will be used to determine observation events, but only records (zero-filled or otherwise) which correspond to a species in `species` will be returned (all others will be omitted). Note that records where `species_id` is NA (generally for 0 counts for presence/absence), will be converted to a list of 0's for the individual species.

Value

Data frame

Examples

```
# Download data (with "core" fields to include 'CommonName')
sample <- nc_data_dl(collection = c("SAMPLE1", "SAMPLE2"), fields_set = "core",
                     username = "sample", info = "nc_example")

# Remove casual observations (i.e. 'AllSpeciesReported' = "No")
library(dplyr) # For filter function
sample <- filter(sample, AllSpeciesReported == "Yes")

# Remove data with "X" ObservationCount (only keep numeric obs)
sample <- filter(sample, ObservationCount != "X")

# Zero fill by all species present
sample_all_zeros <- format_zero_fill(sample)
```

```
# Zero fill only for Canada Goose
goose <- format_zero_fill(sample, species = "230")

# Keep species-specific variables
goose <- format_zero_fill(sample, species = "230", extra_species = "CommonName")

# Keep sampling-event-specific variables
coords <- format_zero_fill(sample, extra_event = c("latitude", "longitude"))

# By species, keeping extra species variables and event variables
goose_coords <- format_zero_fill(sample, species = "230",
                                 extra_species = "CommonName",
                                 extra_event = c("latitude", "longitude"))

# Only return event information
events <- format_zero_fill(sample, fill = NA,
                           extra_event = c("latitude", "longitude"))
```

---

grid_canada *Create grid across Canada*

Description

Create grid across Canada

Usage

```
grid_canada(cell_size = 200, buffer = 500)
```

Arguments

cell_size	Numeric. Size of grid (km) to use when creating grid. If using this grid as input to cosewic_ranges(), should use default COSEWIC grid size of 2.
buffer	Numeric. Extra buffer (km) to add around the outline of Canada before calculating grid.

Value

sf data frame with polygon grid

Examples

```
gc <- grid_canada(200)
gc_buff <- grid_canada(200, buffer = 0)

# Plot to illustrate
```

```
library(ggplot2)
ggplot() +
  geom_sf(data = map_canada()) +
  geom_sf(data = gc, fill = NA) +
  labs(caption = "200km buffer")

ggplot() +
  geom_sf(data = map_canada()) +
  geom_sf(data = gc_buff, fill = NA) +
  labs(caption = "No buffer")
```

---

hofi   *Example house finch data*

Description

An example of house finch data downloaded from NatureCounts

Usage

```
hofi
```

Format

A data frame with 19 rows and 57 variables:

---

map_canada   *Map of Canada*

Description

Wrapper around `rnaturalearth::ne_countries()` to creates a simple features basic map of Canada with CRS 3347 (Statistics Canada Lambert).

Usage

```
map_canada()
```

Value

Sf data frame

Examples

```
map_canada()

plot(map_canada())

library(ggplot2)
ggplot(data = map_canada()) + geom_sf()
```

---

meta   *Metadata*

Description

These functions return metadata codes, names, descriptions, and information associated with the data downloaded from NatureCounts.

Usage

```
meta_country_codes()

meta_statprov_codes()

meta_subnational2_codes()

meta_iba_codes()

meta_bcr_codes()

meta_utm_squares()

meta_species_authority()

meta_species_codes()

meta_species_taxonomy()

meta_collections()

meta_breeding_codes()

meta_project_protocols()

meta_projects()

meta_protocol_types()
```

```
meta_bmde_versions()

meta_bmde_fields(version = "minimum")
```

Arguments

version   Character. BMDE version for which to return fields. NULL returns all versions

Details

Some of these metadata are stored locally and can be updated with the `nc_metadata()` function. Others are downloaded as requested.

Value

Data frame

Functions

- `meta_country_codes()`: Country codes

- `meta_statprov_codes()`: State/Province codes

- `meta_subnational2_codes()`: Subnational2 codes

- `meta_iba_codes()`: Important Bird Area (IBA) codes

- `meta_bcr_codes()`: Bird Conservation Region (BCR) codes

- `meta_utm_squares()`: UTM Square codes

- `meta_species_authority()`: Species taxonomic authorities

- `meta_species_codes()`: Alpha-numeric codes for avian species

- `meta_species_taxonomy()`: Codes and taxonomic information for all species

- `meta_collections()`: Collections names and descriptions

- `meta_breeding_codes()`: Breeding codes and descriptions

- `meta_project_protocols()`: Project protocols

- `meta_projects()`: Projects ids, names, websites, and descriptions

- `meta_protocol_types()`: Protocol types and descriptions

- `meta_bmde_versions()`: Names and descriptions of the available versions of BMDE (Bird Monitoring Data Exchange). These refer to sets of fields/columns which can be downloaded for a given group of data. See `nc_data_dl()` for more details.

- `meta_bmde_fields()`: Fields/columns associated with a particular BMDE (Bird Monitoring Data Exchange) version. See `meta_bmde_versions()` for the different versions available, `meta_collections()` for which version is used by which project, and `nc_data_dl()` for more details on downloading data with a given set of fields/columns.

Examples

```
# Return fields/columns in the 'minimum' version
meta_bmde_fields()

# Retrun fields/columns in the 'core' version
meta_bmde_fields(version = "core")

# Return all possible fields
meta_bmde_fields(version = "extended")
```

---

nc_count   *Download information about NatureCounts collections*

Description

Download the number of records available for different collections filtered by location (if provided). If authorization is provided, the collections are filtered to only those available to the user (unless using show = "all"). Without authorization all collections are returned.

Usage

```
nc_count(
  collections = NULL,
  project_ids = NULL,
  species = NULL,
  years = NULL,
  doy = NULL,
  region = NULL,
  site_type = NULL,
  show = "available",
  username = NULL,
  timeout = 120,
  verbose = TRUE
)
```

Arguments

collections   Character vector. The collection codes from which to download data. NULL
                (default) downloads data from all available collections

project_ids   Character/Numeric vector. The project ids from which to download data.
                First the collections associated with a project_id are determined, and then
                data is downloaded for each collection. If both collections and project_ids
                are supplied, they are combined.

species   Numeric vector. Numeric species ids (see details)

years   Numeric vector. The start/end years of data to download. Can use NA for either
                start or end, or a single value to return data from a single year.

doy	Character/Numeric vector. The start/end day-of-year to download (1-366 or dates that can be converted to day of year). Can use NA for either start or end
region	List. Named list with *one* of the following options: country, statprov, subnational2, iba, bcr, utm_squares, bbox. See details
site_type	Character vector. The type of site to return (e.g., IBA).
show	Character. Either "all" or "available". "all" returns counts from all data sources. "available" only returns counts for data available for the username provided. If no username is provided, defaults to "all".
username	Character vector. Username for <http://naturecounts.ca>. If provided, the user will be prompted for a password. If left NULL, only public collections will be returned.
timeout	Numeric. Number of seconds before connecting to the server times out.
verbose	Logical. Show messages?

Details

The akn_level column describes the level of data access for that collection (see [descriptions online](#)). The access column describes the accessibility of a collection for a given username (or no access if no username supplied). See the section on Access and request_ids for more details.

Value

Data frame

NatureCounts account

All public data is available with a username/password ([sign up](#) for a free NatureCounts account). However, to access private/semi-public projects/collections you must request access. See the Access and request_ids section for more information.

Species ids (species)

Numeric species id codes can determined from the functions [search_species()](#) or [search_species_code()](#). See also the article on [species codes](#) for more information.

Day of Year (doy)

The format for day of year (doy) is fairly flexible and can be a whole number between 1 and 366 or anything recognized by [lubridate-package](#)'s [ymd()](#) function. However, it must have the order of year, month, day. Note that year is ignored when converting to day of year, except that it will result in a 1 day offset for leap years.

Regions (region)

Regions are defined by codes reflecting the country, state/province, subnational (level 2), Important Bird Areas (IBA), and Bird Conservation Regions (BCR) (see [search_region()](#) for codes). They can also be defined by providing specific UTM squares to download or a bounding box area which specifies the min/max longitude and min/max latitude (bbox). See the article on [regional filters](#) for more information.

Access and `request_id`**s**

Access to a data collection is either available as "full" or "by request". Use `nc_count(username = "USER", show = "all")`, to see the accessibility of collections.

"Full" access means that data can be immediately requested directly through the `naturecounts` R package. "By request" means that a request must be submitted online and approved before the data can be downloaded through `naturecounts`.

This means that there are two types of data requests: ones made through this `naturecounts` R package (API requests) and those made through the online Web Request Form (Web requests). Every request (from either method) generates a `request_id` which identifies the filter set and collections requested. Details of all of requests can be reviewed with the `nc_requests()` function.

To download data with "full" access, users can either specify filters, or if they are repeating a download, can use the `request_id` from `nc_requests()`.

Otherwise, if the user doesn't have "full" access, they must supply an approved `request_id` to the `nc_data_dl()` function (e.g., `nc_data_dl(request_id = 152000, username = "USER")`). Use `nc_requests()` to see `request_id`s, filters, and approval status.

Requests for "full" access to additional collections can be made online through the Web Request Form by checking the "Full access?" box in Step 2 of the form.

See Also

`nc_requests()`

Examples

```
# Count all publicly available records:

nc_count()


# Count publicly available records for Manitoba, Canada

nc_count(region = list(statprov = "MB"))


# Count all records for all collections user "sample" has access to
## Not run:
nc_count(username = "sample")

## End(Not run)

# Count records with house finches in Ontario
search_species("house finch")
nc_count(species = 20350, region = list(statprov = "ON"), username = "sample")

# Count all records available in the Christmas Bird Count and Breeding Bird
# Survey collections (regardless of user permissions)
nc_count(collections = c("CBC", "BBS"), show = "all", username = "sample")
```

---

nc_data_dl   *Download NatureCounts data records*

Description

Download data records from various collections filtered by various options. In order to ease the load on the server, note that only **three** of `collections`/`project_ids`, `species`, `years`, `doy`, `region`, and `site_type` can be used in any one request. See the vignette for filtering your data after download for more options: `vignette("filtering_data", package = "naturecounts")`.

Usage

```
nc_data_dl(
  collections = NULL,
  project_ids = NULL,
  species = NULL,
  years = NULL,
  doy = NULL,
  region = NULL,
  site_type = NULL,
  fields_set = "minimum",
  fields = NULL,
  username,
  info = NULL,
  request_id = NULL,
  sql_db = NULL,
  warn = TRUE,
  timeout = 120,
  verbose = TRUE
)
```

Arguments

collections	Character vector. The collection codes from which to download data. NULL (default) downloads data from all available collections
project_ids	Character/Numeric vector. The project ids from which to download data. First the collections associated with a `project_id` are determined, and then data is downloaded for each collection. If both `collections` and `project_ids` are supplied, they are combined.
species	Numeric vector. Numeric species ids (see details)
years	Numeric vector. The start/end years of data to download. Can use NA for either start or end, or a single value to return data from a single year.
doy	Character/Numeric vector. The start/end day-of-year to download (1-366 or dates that can be converted to day of year). Can use NA for either start or end
region	List. Named list with *one* of the following options: country, statprov, subnational2, iba, bcr, utm_squares, bbox. See details

site_type	Character vector. The type of site to return (e.g., IBA).
fields_set	Character. Set of fields/columns to download. See details.
fields	Character vector. If fields_set = custom, which fields/columns to download. See details
username	Character vector. Username for http://naturecounts.ca. If provided, the user will be prompted for a password. If left NULL, only public collections will be returned.
info	Character vector. Short description of reason for the download. E.g., "COSEWIC report", "Impact Assessment Study", "School project", etc. This kind of information helps NatureCounts.ca justify the utility of the database. Required unless resuming/re-downloaded with a request_id.
request_id	Numeric. Specific request id to check or download.
sql_db	Character vector. Name and location of SQLite database to either create or add to
warn	Logical. Interactive warning if request more than 1,000,000 records to download.
timeout	Numeric. Number of seconds before connecting to the server times out.
verbose	Logical. Show messages?

Value

Data frame or connection to SQLite database

NatureCounts account

All public data is available with a username/password (sign up for a free NatureCounts account). However, to access private/semi-public projects/collections you must request access. See the Access and request_ids section for more information.

Species ids (species)

Numeric species id codes can determined from the functions search_species() or search_species_code(). See also the article on species codes for more information.

Day of Year (doy)

The format for day of year (doy) is fairly flexible and can be a whole number between 1 and 366 or anything recognized by lubridate-package's ymd() function. However, it must have the order of year, month, day. Note that year is ignored when converting to day of year, except that it will result in a 1 day offset for leap years.

Regions (region)

Regions are defined by codes reflecting the country, state/province, subnational (level 2), Important Bird Areas (IBA), and Bird Conservation Regions (BCR) (see search_region() for codes). They can also be defined by providing specific UTM squares to download or a bounding box area which specifies the min/max longitude and min/max latitude (bbox). See the article on regional filters for more information.

Data Fields/Columns (`fields_set` **and** `fields`**)**

By default data is downloaded with the `minimum` set of fields/columns. However, for more advanced applications, users may wish to specify which fields/columns to return. The Bird Monitoring Data Exchange (BMDE) schema keeps track of variables used to augment observation data. There are different versions reflecting different collections of variables which can be specified for download in one of four ways:

1. `fields_set` can be a specific shorthand reflecting a BMDE version: `core`, `extended` or `minimum` (default). See `meta_bmde_versions()` to see which BMDE version the shorthand refers to.

2. `fields_set` can be `default` which uses the default BMDE version for a particular collection (note that if you download more than one collection, the field sets will expand to cover all fields/columns in the combined collections)

3. `fields_set` can be the exact BMDE version. See `meta_bmde_versions()` for options.

4. `fields_set` can be `custom` and the `fields` argument can be a character vector specifying the exact fields/columns to return. See `meta_bmde_fields()`) for potential `fields` values.

Note that in all cases there are a set of fields/columns that are *always* returned, no matter what `fields_set` is used.

Access and `request_id`**s**

Access to a data collection is either available as "full" or "by request". Use nc_count(username = "USER", show = "all"), to see the accessibility of collections.

"Full" access means that data can be immediately requested directly through the `naturecounts` R package. "By request" means that a request must be submitted online and approved before the data can be downloaded through `naturecounts`.

This means that there are two types of data requests: ones made through this `naturecounts` R package (API requests) and those made through the online Web Request Form (Web requests). Every request (from either method) generates a `request_id` which identifies the filter set and collections requested. Details of all of requests can be reviewed with the `nc_requests()` function.

To download data with "full" access, users can either specify filters, or if they are repeating a download, can use the request_id from `nc_requests()`.

Otherwise, if the user doesn't have "full" access, they must supply an approved request_id to the `nc_data_dl()` function (e.g., nc_data_dl(request_id = 152000, username = "USER")). Use `nc_requests()` to see request_ids, filters, and approval status.

Requests for "full" access to additional collections can be made online through the Web Request Form by checking the "Full access?" box in Step 2 of the form.

Examples

```
# All observations part of the SAMPLE1 and SAMPLE2 collections
sample <- nc_data_dl(collections = c("SAMPLE1", "SAMPLE2"),
                     username = "sample", info = "nc_example")

# All observations part of project_id 1042 accessible by "testuser"
p1042 <- nc_data_dl(project_ids = 1042, username = "testuser",
```

```
                             info = "nc_example")

   # Black-capped Chickadees (BCCH) in SAMPLE2 collection in 2013
   search_species("black-capped chickadee") # Find the species_id
   bcch <- nc_data_dl(collection = "SAMPLE2", species = 14280, year = 2013,
                      username = "sample", info = "nc_example")

   # All BCCH observations since 2015 accessible to user "sample"
   bcch <- nc_data_dl(species = 14280, years = c(2015, NA), username = "sample",
                      info = "nc_example")

   # All BCCH observations from mid-July to late October in all years for user "sample"
   bcch <- nc_data_dl(species = 14280, doy = c(200, 300), username = "sample",
                      info = "nc_example")

   # All BCCH observations from a specific bounding box for user "sample"
   bcch <- nc_data_dl(species = 14280, username = "sample",
                      region = list(bbox = c(left = -100, bottom = 45,
                                             right = -80, top = 60)),
                      info = "nc_example")

   # All American Bittern observations from user "sample"
   search_species("american bittern")
   bittern <- nc_data_dl(species = 2490, username = "sample", info = "nc_example")

   # Different fields/columns
   bittern <- nc_data_dl(species = 2490, fields_set = "core",
                         username = "sample", info = "nc_example")

   bittern <- nc_data_dl(species = 2490, fields_set = "custom",
                         fields = c("Locality", "AllSpeciesReported"),
                         username = "sample", info = "nc_example")

   ## Not run:
   # All collections by request id

   # Specific collection by request id
   my_data <- nc_data_dl(collections = "ABATLAS1",
                         request_id = 000000, username = "USER",
                         info = "MY REASON")

   ## End(Not run)
```

---

nc_metadata   *Update NatureCounts metadata files*

Description

Updates the local copies of meta data used by the package.

Usage

```
nc_metadata(force = FALSE, utm = FALSE, verbose = TRUE)
```

Arguments

force	Logical. Force update even if the remote version matches local?
utm	Logical. Update meta_utm_squares() as well? **WARNING**: This is a large and time consuming download!
verbose	Logical. Show progress messages?

Examples

```
nc_metadata()
```

---

nc_permissions   *Download list of accessible collections*

Description

Returns a list of collections accessible by 'username'.

Usage

```
nc_permissions(username = NULL, timeout = 60)
```

Arguments

username	Character vector. Username for http://naturecounts.ca. If provided, the user will be prompted for a password. If left NULL, only public collections will be returned.
timeout	Numeric. Number of seconds before connecting to the server times out.

NatureCounts account

All public data is available with a username/password (sign up for a free NatureCounts account). However, to access private/semi-public projects/collections you must request access. See the Access and request_ids section for more information.

Access and `request_id`**s**

Access to a data collection is either available as "full" or "by request". Use `nc_count(username =` `"USER", show = "all")`, to see the accessibility of collections.

"Full" access means that data can be immediately requested directly through the `naturecounts` R package. "By request" means that a request must be submitted online and approved before the data can be downloaded through `naturecounts`.

This means that there are two types of data requests: ones made through this `naturecounts` R package (API requests) and those made through the online Web Request Form (Web requests). Every request (from either method) generates a `request_id` which identifies the filter set and collections requested. Details of all of requests can be reviewed with the `nc_requests()` function.

To download data with "full" access, users can either specify filters, or if they are repeating a download, can use the `request_id` from `nc_requests()`.

Otherwise, if the user doesn't have "full" access, they must supply an approved `request_id` to the `nc_data_dl()` function (e.g., `nc_data_dl(request_id = 152000, username = "USER")`). Use `nc_requests()` to see `request_id`s, filters, and approval status.

Requests for "full" access to additional collections can be made online through the Web Request Form by checking the "Full access?" box in Step 2 of the form.

Examples

```
nc_permissions()
nc_permissions(username = "sample")
```

`nc_query_table`   *Custom table queries*

Description

Generate custom table queries with the table name and filter arguments.

Usage

```
nc_query_table(
  table = NULL,
  ...,
  username = NULL,
  timeout = 120,
  verbose = FALSE
)
```

Arguments

`table`	Character. Table to query (see details)
`...`	Name/value pairs for custom queries/filters (see details)
`username`	Character vector. Username for <http://naturecounts.ca>. If provided, the user will be prompted for a password. If left NULL, only public collections will be returned.
`timeout`	Numeric. Number of seconds before connecting to the server times out.
`verbose`	Logical. Show messages?

Details

nc_query_table(username = "sample") for available options

Value

data.frame()

Examples

```
# What tables are available? What 'filters' do they take? Are any 'required'?

nc_query_table(username = "sample")

# Query the bmdefilter_bad_dates table

d <- nc_query_table(table = "bmde_filter_bad_dates", username = "sample")
head(d)

# Filter our query
d <- nc_query_table(table = "bmde_filter_bad_dates",
                    SiteCode = "DMBO", username = "sample")
d

# Filter our query
d <- nc_query_table(table = "bmde_filter_bad_dates",
                    species_id = 15770, username = "sample")

# Want more than one species? Either filter after, or combine two queries

# Filter after
library(dplyr)
d <- nc_query_table(table = "bmde_filter_bad_dates", username = "sample")
d <- filter(d, species_id %in% c(15770, 9750))

# Combine two queries
d1 <- nc_query_table(table = "bmde_filter_bad_dates",
                     species_id = 15770, username = "sample")
d2 <- nc_query_table(table = "bmde_filter_bad_dates",
                     species_id = 9750, username = "sample")
d <- rbind(d1, d2)
```

---

nc_remove_cache   *Remove in-memory cache*

Description

All server queries are cached for four hours to reduce server load. You can reset the cache at any time by either restarting your R session or running `nc_remove_cache()`.

Usage

```
nc_remove_cache()
```

Value

TRUE if it worked

Examples

```
nc_remove_cache()
```

---

nc_requests   *Check on status of data requests*

Description

List pending or completed data requests for a given user.

Usage

```
nc_requests(request_id = NULL, type = "web", username)
```

Arguments

request_id	Numeric. Specific request id to check or download.
type	Character One of "web", "api", or "all" specifying which types of request to return (defaults to "web").
username	Character vector. Username for <http://naturecounts.ca>. If provided, the user will be prompted for a password. If left NULL, only public collections will be returned.

Value

data frame

Access and `request_id`**s**

Access to a data collection is either available as "full" or "by request". Use `nc_count(username =` `"USER", show = "all")`, to see the accessibility of collections.

"Full" access means that data can be immediately requested directly through the `naturecounts` R package. "By request" means that a request must be [submitted online](#) and approved before the data can be downloaded through `naturecounts`.

This means that there are two types of data requests: ones made through this `naturecounts` R package (API requests) and those made through the online [Web Request Form](#) (Web requests). Every request (from either method) generates a `request_id` which identifies the filter set and collections requested. Details of all of requests can be reviewed with the [`nc_requests()`](#) function.

To download data with "full" access, users can either specify filters, or if they are repeating a download, can use the `request_id` from [`nc_requests()`](#).

Otherwise, if the user doesn't have "full" access, they must supply an approved `request_id` to the [`nc_data_dl()`](#) function (e.g., `nc_data_dl(request_id = 152000, username = "USER")`). Use [`nc_requests()`](#) to see `request_id`s, filters, and approval status.

Requests for "full" access to additional collections can be made online through the [Web Request Form](#) by checking the "Full access?" box in Step 2 of the form.

Examples

```
nc_requests(username = "sample")
nc_requests(request_id = 152446, username = "sample")
```

---

search_region	*Find country, state/province, subnational2, IBA, or BCR codes*

Description

Search for the correct codes to identify countries, states/provinces, subnational2 areas, Important Bird Areas (IBA), or Bird Conservation Regions (BCR). These are then used in the [`nc_data_dl`](#)() and [`nc_count`](#)() functions.

Usage

```
search_region(name = NULL, type = "country")
```

Arguments

name	Character. The location name to search for
type	Character. One of "country", "statprov", "subnational2", "iba", or "bcr". The type of information to return.

Details

`region_search()` is deprecated in favour of `search_region()`

Value

A data frame with the relevant codes and other information

Examples

```
search_region("Mexico", type = "country")   # MX

search_region("Yucatan", type = "statprov") # Yucatán
search_region("Alberta", type = "statprov") # AB

search_region("Edmonton", type = "subnational2") # CA.AB.11
search_region("Brandon", type = "subnational2")  # CA.MB.07

search_region("hays reservoir", type = "iba") # AB075
search_region("rainforest", type = "bcr")     # 5


# Show all codes
search_region(type = "country")
search_region(type = "statprov")
search_region(type = "subnational2")
search_region(type = "iba")
search_region(type = "bcr")


# Using the codes
nc_count(region = list(statprov = "AB"), years = 2010)
```

---

search_species   *Find species codes*

Description

Find species id codes by searching for scientific, English and French species names.

Usage

```
search_species(name = NULL, show = "names", authority = NULL)
```

Arguments

name	Character. The species name to search for
show	Character. Either "all" or "names" (default). Whether to return all taxonomic information or only a subset with species names
authority	Character. If not NULL (default), return the alphanumeric code associated with avian species for this taxonomic authority.

Details

species_search() is deprecated in favour of search_species()

Value

Data frame of species ids and taxonomic information

Examples

```
# Show all ids
search_species()

search_species("chickadee")
search_species("black-capped chickadee")

# Add alphanumeric code for BSCDATA authority
search_species("black-capped chickadee", authority = "BSCDATA")

# Show all taxonomic information
search_species("black-capped chickadee", show = "all")

# Using the codes
nc_count(species = 14280)
```

---

search_species_code   *Search for bird species id codes by alphanumeric codes*

Description

This is an advanced function for returning all Bird-related species id codes based on the various alphanumeric codes used by different authorities.

Usage

```
search_species_code(code = NULL, authority = "BSCDATA", results = "all")
```

Arguments

code   Vector. Character or numeric code indicating a species for a given authority.

authority   Character. The authority to compare codes against (defaults to "BSCDATA")

results   Character. "all" returns codes for all related species (including subspecies and main species). "exact" returns only the code for exact species indicated by the code.

Details

species_code_search() is deprecated in favour of search_species_code()

Species ids returned reflect both species and sub-species levels.

Value

A data frame of numeric species id codes and names

Examples

```
# Show all ids
search_species_code()

# Get all species ids for house finches
search_species_code("HOFI")

# Get all species ids for Dark-eyed Juncos
search_species_code("DEJU")

# Get all species ids related to Yellow-rumped Warbler (Myrtle)
# NOTE! This includes Audubon's and the main, Yellow-rumped Warbler species
search_species_code("MYWA")

# Get ONLY specific id related to Yellow-rumped Warbler (Myrtle)
search_species_code("MYWA", results = "exact")

# Use the Christmas Bird Count authority
search_species_code(11609, authority = "CBC")

# Look in more than one authority (note that the code only needs to match on
# of the authorities)
search_species_code("MYWA", authority = c("BCMA", "CBC"))
```

Index